Penerapan *B-Tree* dalam Sistem Berkas (*File System*)

Hani'ah Wafa - 13516053
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516053@std.stei.itb.ac.id

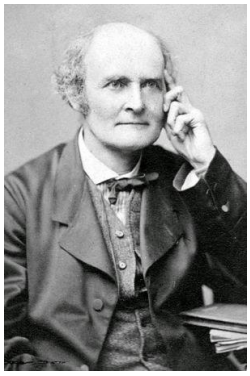
Abstrak

Semakin hari ukuran atau jumlah data yang dapat disimpan di dalam sebuah komputer maupun penyimpanan eksternal semakin bertambah besar. Untuk itu diperlukan sebuah sistem yang membuat penyimpanan data-data tersebut menjadi lebih tersturktur. Penyimpanan yang terstruktur itu tentu saja sangat diperlukan agar nantinya pencarian berkas atau arsip dapat dilakukan dengan efisien dan tanpa membutuhkan waktu yang lama.

Salah satu metode yang dapat digunakan dalam membuat struktur sistem berkas yang kemudian juga akan dibahas dalam makalah ini adalah dengan menggunakan struktur data pohon. Struktur data berupa pohon yang digunakan untuk menggambarkan struktur penyimpanan data ini kemudian disebut juga dengan *directory tree*.

Kata Kunci : pohon, sistem berkas, *directory*, *directory tree*.

I. PENDAHULUAN



Gambar 1.1 : Arthur Cayley
Sumber : https://www.wikitree.com/wiki/Cayley-44

Pohon (*tree*) adalah struktur matematika yang dapat digambarkan baik sebagai graf maupun sebagai struktur data. Kedua penggambaran tersebut ekivalen karena struktur data pohon yang dibuat tidak hanya berupa himpunan elemen-elemen, tapi juga koneksi atau penghubung antar elemen yang membuatnya menjadi sebuah graf pohon.

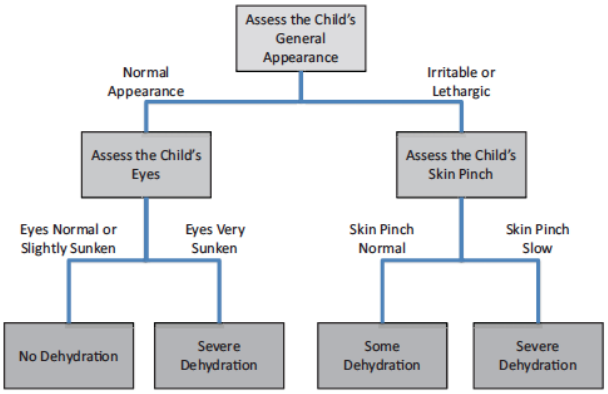
Pertama kali dipelajari oleh Arthur Cayley, seorang matematikawan Inggris pada tahun 1857, saat itu pohon digunakan untuk menghitung jumlah senyawa kimia.

Pohon merupakan graf tidak berarah yang juga tidak memiliki sirkuit. Atau dengan kata lain dapat dikatakan bahwa pohon merupakan graf yang sederhana, tidak berarah, terhubung, dan tidak memiliki sirkuit. Pohon dengan n buah simpul pasti akan memiliki n-1 buah sisi dan semua pohon merupakan graf bipartide.

Pohon juga dapat diklasifikasikan lagi menjadi beberapa jenis seperti, pohon merentang, pohon berakar, pohon n-ary, pohon seimbang, dan sebagainya. Secara singkat, pohon merentang dari sebuah graf terhubung, adalah upagraf merentang yang berupa pohon. Pohon berakar adalah pohon yang sebuah simplnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga terbentuk graf berarah, lalu simpul lainnya akan diperlakukan sebagai anak. Sedangkan pohon n-ary adalah pohon berakar yang tiap simpulnya memiliki paling banyak n buah anak. Pohon n-ary dikatakan sebagai pohon yang *full* atau lengkap jika setiap simpul dalam pohon tersebut kecuali daun memiliki tepat n buah anak. Kemudian sebuah pohon dikatakan sebagai pohon seimbang jika subpohonnya merupakan pohon seimbang dan perbedaan tinggi antar subpohonnya maksimal adalah satu.

Selain untuk menghitung jumlah senyawa kimia seperti yang dilakukan oleh Cayley, pohon dapat dimanfaatkan untuk berbagai kebutuhan lainnya. Contoh lain dari penerapan pohon salah satunya adalah untuk menggambarkan silsilah keluarga. Lalu pohon merentang juga dapat dimanfaatkan untuk mencari bobot terkecil yang diperlukan untuk melewati seluruh simpul yang ada pada suatu graf.

Tidak hanya itu, pengaplikasian pohon keputusan juga dapat kita gunakan untuk membuat keputusan. Seperti contohnya saat seorang dokter akan menentukan penyakit yang diderita oleh seorang pasien berdasarkan gelaja yang dideritanya. Pohon juga dapat dimanfaatkan untuk menggambarkan sistem pertandingan, dan seperti yang akan dibahas lebih lanjut yaitu digunakan menjadi struktur data dalam sistem berkas.



Gambar 1.2 : Pohon keputusan untuk menetukan jenis dehidrasi anak berdasarkan gejala yang tampak
Sumber : http://www.futurity.org/childrens-health-dehydration-986942

II. SISTEM BERKAS (*FILE SYSTEM*)

File system terbagi menjadi dua kata, yaitu *file* dan *system*. Dimana *file* itu sendiri dapat diartikan sebagai sekumpulan data. Sedangkan struktur atau aturan-aturan yang digunakan untuk memanajemen kmpulan data tersebut lah yang disebut sebagai *file system*.

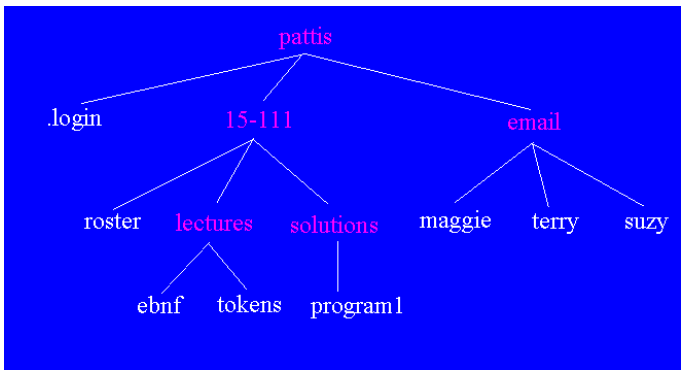
Sistem berkas atau *file system* digunakan untuk mengatur bagaimana sebuah data disimpan dan diakses kembali. Tanpa sistem berkas, informasi yang tersimpan akan terkumpul dalam sebuah kumpulan data yang sangat besar tanpa dapat diketahui dimana sebuah informasi akan berakhir dan dimana informasi yang selanjutnya berawal.

Dengan membagi kumpulan data tersebut menjadi beberapa bagian dan memberi nama pada setiap data yang ada, maka informasi-informasi yang kita miliki akan lebih mudah untuk dipisahkan dan dikenali.

III. POHON DIREKTORI (*DIRECTORY TREE*)

Pohon direktori (*directory tree*) adalah hirarki direktori yang terdiri dari sebuah direktori yang disebut sebagai orang tua (*parent directory*) atau *top level directory* dan sisanya sebagai subdirektori. Berdasarkan definisi tersebut maka pohon direktori termasuk sebagai pohon berakar dengan n-ary.

Nama pohon direktori itu sendiri diambil karena fakta bahwa diagram yang digambarkan oleh struktur ini akan menyerupai pohon terbalik, dimana setiap simpul merupakan sebuah file atau folder yang cabangnya merupakan sebuah file atau folder juga yang kemudian mungkin akan bercabang lagi. Pada pengaplikasiannya, jenis pohon yang digunakan sebagai pohon direktori ini merupakan *B-Tree*.



Gambar 3.1 : Pohon direktori, dengaan nama folder ditulis dengan warna pink dan nama file ditulis dengan warna putih.
Sumber : http://www.linfo.org/directory_tree.html

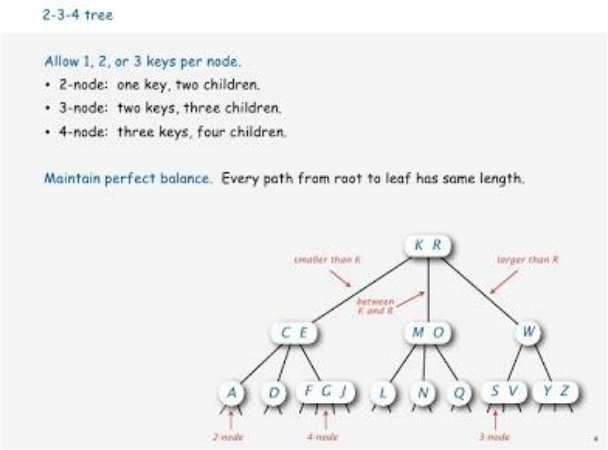
IV. B-TREE SEBAGAI POHON DIREKTORI

A. Definisi B-Tree

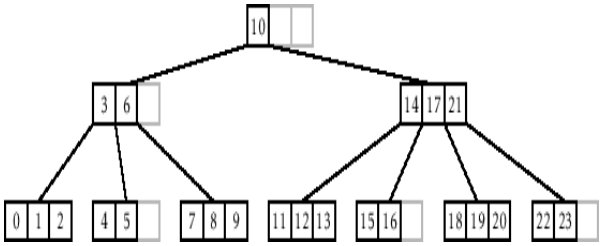
B-Tree adalah struktur data pohon seimbang yang merupakan generalisasi dari pohon biner dan merupakan struktur data yang efisien dalam memodelkan memori eksternal. Struktur data B-Tree juga membuat kita dapat melakukan berbagi operasi dengan waktu asimptotik berupa logaritma (O (log_n)).

B. Struktur B-Tree

Tidak seperti pohon biner yang hanya dapat memiliki maksimal 2 buah anak pada tiap simpulnya, pohon B seimbang dapat memiliki B buah kunci dan B buah anak di setiap simpulnya. Kunci pada pohon B seimbang disimpan dalam urutan tidak menurun. Anak yang dimilki oleh suatu kunci akan menjadi akar dari subpohon dengan simpul yang kuncinya lebih kecil atau sama dengan kunci tersebut, namun lebih besar dari kunci sebelumnya (di sebelah kirinya). Sebuah simpul juga memiliki tambahan anak terkanan yang akan menjadi akar dari subpohon dengan nilai kunci yang lebih besar dari kunci manapun yang ada di tiap simpul.



Gambar 4.1 : Contoh B-Tree
Sumber :
https://www.slideshare.net/mobile/sriprasanna/balanced-trees-1941548



Gambar 4.2 : B-Tree dengan B=2.
Sumber : http://opendatastructures.org/versions/edition-0.1g/ods-python/14_2_B_Trees.html

C. Tinggi Pohon Seimbang

Untuk semua bilangan bulat B≥2, sebuah B-Tree adalah pohon yang seluruh daunnya memiliki kedalaman yang sama, dan setiap simpul dalam kecuali akar memiliki paling sedikit B buah anak dan paling banyak 2B buah anak.

Jika tinggi suatu B-Tree adalah h dan banyaknya daun dalam pohon tersebut adalah l, maka akan berlaku :

$$2B^{h-1} \leq l \leq 2\,(2B)^{h-1}$$

Sehingga dengan mengaplikasikan logaritma akan didapatkan :

$$h \leq \frac{\log l - 1}{\log B} + 1$$

$$\leq \frac{\log l}{\log B} + 1$$

$$= log_B l + 1$$

Dari persamaan tersebut dapat kita simpulkan bahwa tinggi dari sebuah pohon B akan ekivalen dengan logaritma dengan basis B dari jumlah daunnya.

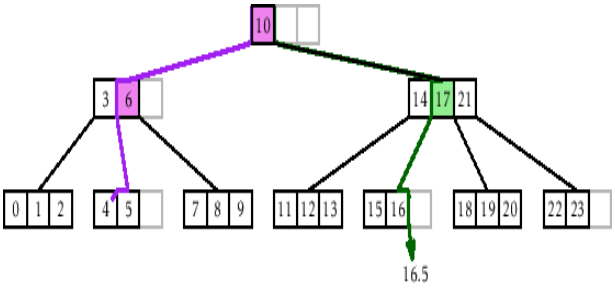
V. OPERASI DALAM *B-TREE*

Ada beberapa jenis operasi yang dapat dilakukan pada sebuah B-Tree, beberapa diantaranya yaitu pencarian (*searching*), penambahan (*add*), dan penghapusan (*removing/deleting*).

A. Pencarian (*Searching*)

Algoritma pencarian yang digunakan intinya adalah dengan menggeneralisasikan skema pencarian pohon biner (*binary search tree*). Pencarian akan dimulai dari akar pohon, lalu menentukan ke anak atau cabang yang manakah pencarian tersebut akan dilanjutkan.

Lebih spesifiknya, misalkan kita sedang mencari suatu bilangan x. Saat kita telah menemukan x pada suatu simpul, maka pencarian akan berakhir. Namun jika tidak, kita akan mencari suatu bilangan i terkecil dimana i > x, kemudian melanjutkan pencarian di subpohon dengan akar i. Jika tidak ada nilai i dimana i > x, maka pencarian akan dilakukan di subpohon yang merupakan anak terkanannya.



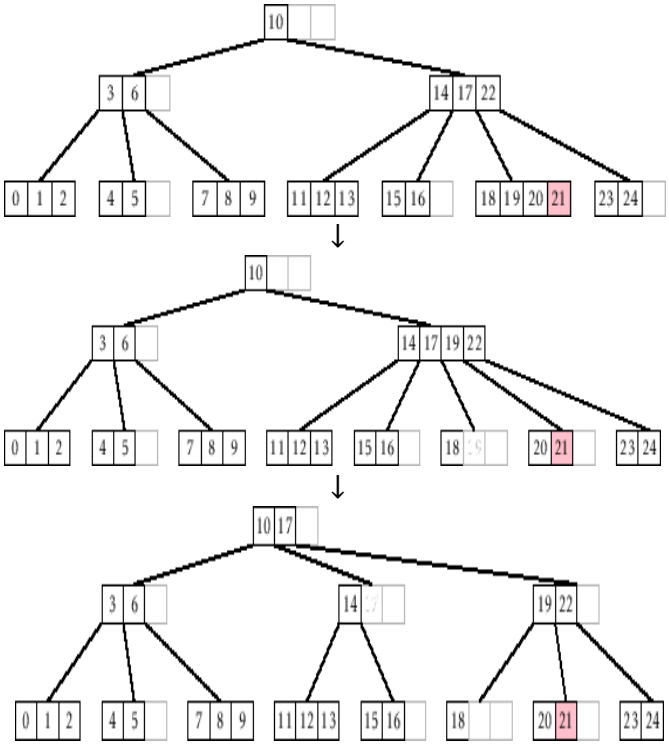
Gambar 5.1 : Pencarian bilangan 4 yang berhasil dan pencarian bilangan 16,5 yang tidak berhasil.
Sumber : http://opendatastructures.org/versions/edition-0.1g/ods-python/14_2_B_Trees.html

B. Penambahan (*Add*)

Perbedaan penting yang terdapat pada struktur pohon biner dan struktur pohon B adalah pada pohon B, sebuah simpul tidak menyimpan *pointer* yang menunjuk pada orang tuanya. Sehingga operasi penambahan dan penghapusan pada pohon B akan paling mudah jika dilakukan secara rekursif.

Pada setiap proses penambahan maupun pengurangan pohon B, kita harus selalu menjaga keseimbangan pohon tersebut. Penjagaan keseimbangan ini nantinya akan

dilakukan dengan cara membagi atau memecah sebuah simpul.

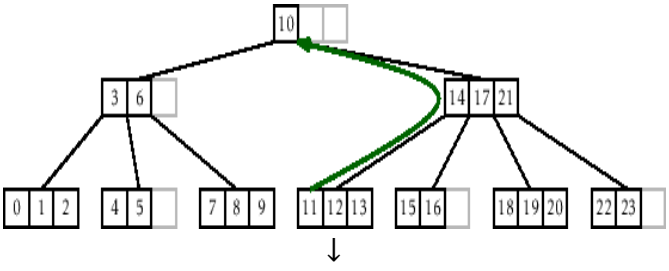


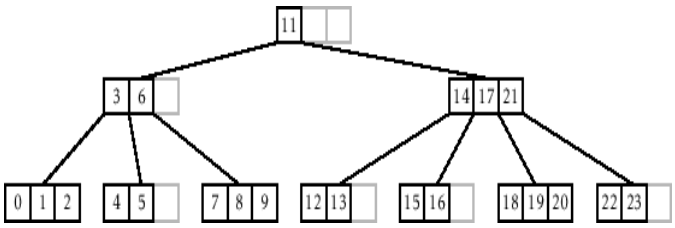
Gambar 5.2 : Proses penambahan bilangan 21 ke pohon B dengan memecah suatu simpul.
Sumber : http://opendatastructures.org/versions/edition-0.1g/ods-python/14_2_B_Trees.html

C. Penghapusan (*Removing/Deleting*)

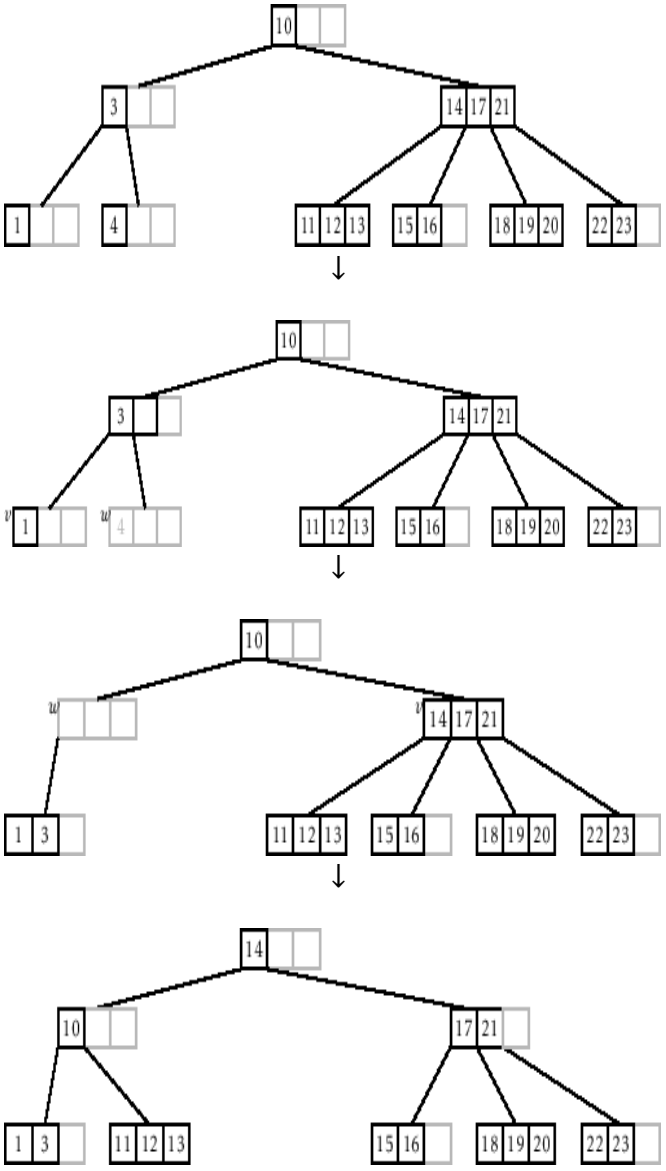
Penghapusan juga dapat dilakukan pada *B-Tree*, sekali lagi seperti halnya penambahan, penghapusan ini juga akan paling mudah jika dilakukan secara rekursif. Saat melakukan penghapusan, kita jugadiharuskan untuk tetap menjaga keseimbangan pohon B dan memerhatikan kasus-kasus khusus yang mungkin terjadi.

Salah satu contoh kasus adalah ketika penghapusan akan mengurangi jumlah kunci, sehingga kuncinya menjadi lebih sedikit dari batas minimum jumlah kunci yang seharusnya. Saat itu terjadi, maka harus diatasi dengan mengkombinasikan beberapa simpul dan memungkinkan untuk mengurangi ketinggian pohon.





Gambar 5.3 : Proses penghapusan bilangan 10 pada pohon B
Sumber : http://opendatastructures.org/versions/edition-0.1g/ods-python/14_2_B_Trees.html



Gambar 5.4 : Proses penghapusan bilangan 4 pada pohon B
Sumber : http://opendatastructures.org/versions/edition-0.1g/ods-python/14_2_B_Trees.html

VI. BTRFS SEBAGAI SISTEM BERKAS PADA SISTEM OPERASI LINUX

Setiap sistem operasi pasti memiliki sistem berkas-nya sendiri. Baik windows, linux, atau sistem operasi yang lain mungkin memiliki sitem file yang berbeda. Tentunya file sistem yang digunakan tersebut akan memiliki kelebihan dan kekrangannya tersendiri.

Pada sistem operasi windows kita sudah mengenal ada beberapa sitem file seperti FAT (*File Allocation Table*), baik itu FAT12, FAT 16, dan sebagainya. Kemudian ada juga NTFS (*New Technology File System*), mulai dari versi 1.0, versi 1.1, dan seterusnya. Sedangkan sistem berkas yang digunakan dalam sistem operasi linux yaitu Ext2, Ext3, dan Ext4.

Lalu dimanakah pengaplikasian B-tree digunakan dalam sistem berkas? Ternyata selain sitem file yang telah disebutkan di atas, ada pula sistem berkas pada sistem operasi linux yang menggunakan pengaplikasian dari pohon B dan baru mulai dikembangkan pada tahun 2007, yaitu Btrfs (*B-tree file system*).

Btrfs (B-tree file sytem / butter FS / better FS) adalah sistem berkas yang dikembangkan oleh Chris Mason dan berada di bawah lisensi *General Public Lisence* (GPL). Btrfs dikembangkan untuk menangani kekurangan yang terdapat pada sistem berkas Ext. Seperti yang dilansir dari laman https://btrfs.wiki.kernel.org/index.php/Main_Page Btrfs adalah sistem berkas yang stabil.

Gambar 6.1 : Chris Mason, pengembang sistem berkas Btrfs
Sumber : https://lwn.net/Articles/465180/

Dengan Btrfs ternyata membuat linux dapat lebih mengatur tempat penyimpanan data yang ada. Mengatur yang dimaksud dalam hal ini bukan hanya sekedar mengatur dalam hal pengalamatan saja. Namun juga dapat melakukan administrasi dan pengelolaan tempat penyimpanan dengan tampilan yang lebih bersih sehingga pengguna dapat melihat apa yang sedang digunakan atau dikerjakan.

Beberapa perbedaan antara Btrfs dan Ext terletak pada penyimpanan maksimal, waktu perputaran disk, kompresi transparan, enkripsi transparan, dan duplikasi data. Dikatakan pula bahwa Btrfs adalah sistem berkas baru dengan fitur yang canggih mirip dengan Sun / Oracle's excellent ZFS. Fitur-fitur yang dimaksud adalah *snapshot, multi-disk striping, multi disk-mirroring, checksum, incremental backup*, dan *on-the-fly compression*, yang dapat memberikan peningkatan kinerja yang signifikan dan menghemat ruang.

VI. MENGAPA *B-TREE*?

Pada dasarnya, struktur data pohon seimbang pada umumnya juga dapat digunakan untuk menjadi sistem direktori file. Lalu mengapa digunakan *B-Tree*?

Seperti yang telah dijelaskan sebelumnya, saat melakukan sebuah operasi pada pohon B akan memiliki kasus terburuk dengan ketinggian O ($log_B l$), dengan B adalah jumlah anak minimal yang harus dimiliki oleh setiap simpul dan l adalah jumlah daun yang dimiliki pohon B. Jumlah anak atau cabang yang dapat dimiliki oleh B-Tree bisa lebih banyak dari jenis struktur data pohon seimbang yang lainnya, maka basis logaritma yang digunakan untuk menghitung ketinggian pohon pun akan menjadi semakin besar. Akibatnya ketinggian pohon dan jumlah simpul yang dikunjungi pada saat melakukan suatu operasi akan menjadi lebih sedikit. Ini berarti waktu yang diperlukan untuk menyelesaikan operasi tersebut pun lebih sedikit.

Meskipun banyaknya percabangan pada pohon B tersebut tidak berdampak pada kasus terburuk ketinggian secara asimptotik, tapi pada kenyatannya ketinggian pohon B akan lebih kecil dibanding ketinggian jenis pohon lain yang memiliki ketinggian asimptotik yang sama.

VII. KESIMPULAN

Setiap jenis penyimpanan data pasti menggunakan suatu sistem berkas agar data yang disimpan akan lebih mudah untuk dapat dikenali dan kemudian diakses kembali. Semakin efisien dan semakin cepat waktu yang dibutuhkan oleh suatu sistem berkas untuk melakukan sebuah operasi tentu akan menjadikan sistem berkas tersebut menjadi lebih baik.

Salah satu sistem berkas dengan waktu ekseskusi yang relatif lebih cepat adalah sistem berkas yang mengaplikasikan struktur data pohon B. Karena kasus terburuk pada operasi yang dilakukan pada sebuah pohon B adalah sebanyak O ($log_B l$). Dimana tidak seperti pohon biner dengan B=2, pohon B bisa membuat memiliki nilai B yang lebih besar dari struktur data pohon seimbang lainnya.

REFERENSI

[1] *Special Trees*
https://www.cs.cmu.edu/~pattis/15-1XX/15-200/lectures/specialtrees/index.html
Diakses pada 1 Desember 2017.

[2] Pohon pada Matematika Diskrit
http://rizaxxi.blogspot.co.id/2015/07/pohon-pada-matematika-diskrit.html?m=1
Diakses pada 1 Desember 2017.

[3] *Directory Tree Definition*
http://www.linfo.org/directory_tree.html
Diakses pada 1 Desember 2017.

[4] *Tree*
http://mathworld.wolfram.com/Tree.html
Diakses pada 1 Desember 2017.

[5] *B-Trees : Balanced Tree Data Structure*
https://www.bluerwhite.org/btree/
Diakses pada 2 Desember 2017.

[6] *Balanced Trees*
https://www.slideshare.net/mobile/sriprasanna/balanced-trees-1941548
Diakses pada 2 Desember 2017.

[7] *B-Trees*
http://opendatastructures.org/versions/edition-0.1g/ods-python/14_2_B_Trees.html
Diakses pada 2 Desember 2017.

[8] *File System*
https://wiki.archlinux.org/index.php/file_systems
Diakses pada 2 Desember 2017.

[9] *File System* di Windows dan Linux
https:/www.google.co.id/amp/s/dhanz3rd.wordpress.com/2010/12/14/file-system-di-windows-dan-linux/amp/
Diakses pada 2 Desember 2017.

[10] Jenis-Jenis Partisi di Linux dan Penjelasannya
https:/www.google.co.id/amp/s/gilangsuleman.wordpress.com/2013/10/21/jenis-jenis-partisi-di-linux-dan-penjelasannya/amp/
Diakses pada 2 Desember 2017.

[11] *Brtfs filesystem* di ubuntu
https://www.google.co.id/amp/s/blackstreetnight.wordpress.com/2013/06/20/btrfs-filesystem-di-ubuntu/amp/
Diakses pada 2 Desember 2017.

[12] *Brtfs*
https://btrfs.wiki.kernel.org/index.php/Main_Page
Diakses pada 2 Desember 2017.

[13] Apa itu *File System*?
gaptex.com/apa-itu-file-system
Diakses pada 2 Desember 2017.

[14] Sistem Berkas (*File System*)
http://saveasnote.blogspot.co.id/2013/06/sistem-berkas-file-system.html?m=1
Diakses pada 2 Desember 2107.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

Hani'ah Wafa - 13516053